US-China Education Review A 9 (2012) 796-803
Earlier title: US-China Education Review, ISSN 1548-6613

Microworlds: Building Powerful Ideas in the Secondary School

Craig William Jenkins
University of Wales, Wales, UK

In the 1960s, the MIT (Massachusetts Institute of Technology) developed a programming language called LOGO. Underpinning this invention was a profound new philosophy of how learners learn. This paper reviews research in the area and asks how one notion in particular, that of a microworld, may be used by secondary school educators to build powerful ideas in STEM (science, technology, engineering, and mathematics) subjects.

Keywords: microworlds, programming, STEM (science, technology, engineering, and mathematics), constructionism, education

Theories of Knowing

This paper examines the microworld as a tool for acquiring powerful ideas in secondary education and explores their potential role in making relevant conceptual learning accessible through practical, constructionist approaches.

In line with this aim, the paper is split into three main sections: The first section looks at the underlying educational theory behind microworlds in order to set up the rest of the paper; The second section critically examines the notion of a microworld in order to draw out the characteristics of a microworlds approach to learning; Finally, the paper ends with a real-world example of a microworld that is designed to build key, powerful ideas within a STEM (science, technology, engineering, and mathematics) domain of knowledge.

To begin to understand the educational theory behind microworlds, a good starting point is to consider the ways in which learners interact with educational technology. In 1980, Robert Taylor (1980) provided a useful framework for understanding such interactions. For Taylor, the part that a computer plays in each interaction can be understood in terms of three roles: “tutor”, “tool”, and “tutee”.

The first type of interaction identified by Taylor is that where the computer teaches the learner, taking the role of tutor. The computer presents subject content to the learner and ascertains the learner’s understanding by providing a test and evaluating the response. Taylor’s (1980) vision of tutor-mode computing is expressed as follows:

> (T)he computer tutor keeps complete records on each student being tutored; it has at its disposal a wide range of subject detail it can present; and it has an extensive and flexible way to test and then lead the student through the material. (p. 3)

The second mode of computing identified by Taylor is where the computer is used as a tool by the learner. A wide variety of time-consuming, low-order tasks can be implemented by the computer quickly and easily, thus enabling the learner to spend their time carrying out more useful activities: “necessary but routine clerical

Craig William Jenkins, Ph.D. candidate, School of Education, University of Wales.

tasks of a tedious mechanical kind (are transferred) to the computer" (Taylor, 1980). And tool-mode computing, for Taylor, has useful applications in many areas of school life:

> (T)he tedious recopying of edited manuscripts of texts or even music can be relegated to the computer through word or musical notation processing software; the laborious drawing of numerous intermediate frames for animated cartoons can be turned over to the computer through graphics software. (Taylor, 1980)

Finally, Taylor's third mode of computing in the classroom is where the computer takes on the role of the tutee. This involves the student programming the computer and learning to "talk to the computer in a language it understands" (Taylor, 1980). Programming using a computer is an integral skill for Taylor, serving as a means to "enable the child to link his or her experience to the deep, fundamental mathematical ideas we most want children to learn" (Taylor, 1980, p. 7).

Taylor was clear that it is the latter mode, where the learner programs the computer, which confers the greatest educational benefit for the learner. A microworlds approach to learning provides a quintessential example of this type of interaction. Traditionally, however, it is the first mode of computing that has dominated the classroom.

The tutor-mode of computing is aligned with what has been termed the "instructionist" approach to educational technology. In a speech to a conference of educators in Japan, Papert (1980b) explained this approach:

> Instructionism is the theory that says, "To get better education, we must improve instruction. And if we're going to use computers, we'll make the computers do the instruction." And that leads into the whole idea of computer-aided instruction. (Papert, 1980b)

More recently, in *The Children's Machine*, Papert (1993) used Paulo Freire's criticism of instructionism to illustrate the epistemological connotations of such a model:

> Paulo Freire expresses the criticism most vividly in his description of school as following a "banking model" in which information is deposited in the child's mind like money in a savings account. (Papert, 1993, p. 14)

Put simply, instructionism views the learner as an antenna to receive transmitted knowledge: The mind is a "vessel to be filled" (Papert, 1993). The transmitter of knowledge is the teacher, or in the case of educational technology, the computer. This epistemological model has, unsurprisingly, given rise to a proliferation of software built around the idea of drill-and-practice. The microworlds approach to learning, on the other hand, is premised on a very different theory of knowing.

Jean Piaget, a self-termed genetic epistemologist, rejected such an "instructionist" theory of knowing. As Boden (1994) pointed out, for Piaget, "Knowledge is… actively constructed, by a dialectical process of assimilation and accommodation" (p. xi). Learners fit external input into their existing mental models through a process of assimilation. In turn, learners change their mental models based on this input through a process of accommodation (Boden, 1994). For Piaget, learning was not about how efficiently a student receives transmitted knowledge. Instead, he proposed that learners are active constructors of knowledge, not passive recipients. Put simply, learners "make" knowledge, they do not "receive" it. This fundamentally different theory of knowing has been coined "constructivism".

The quintessential microworld of Turtle Graphics, created by Papert and his team at the MIT (Massachusetts Institute of Technology), is rooted in this constructivist theory of knowing. More specifically,

Papert was concerned with a particular mode of "constructivism" (with a "v") that he termed "constructionism" (with a "n"). In *Situating Constructionism*, Papert (1991) made this very important distinction:

> Constructionism—the "n" word as opposed to the "v" word—share constructivism's connotation of learning as "building knowledge structures" (and) then adds the idea that it happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sandcastle on the beach or a theory of the universe.

In this way, constructionism (with a "n") entails constructivism (with a "v") since both theories of knowing situate learners as active constructors of their own knowledge. What constructionism adds, then, is that knowledge is best constructed when learners are making meaningful things. It is for this reason that Papert's theory is often summed up by the phrases "learning by making" and "learning by designing". As Mitchel Resnick (2002, p. 33) explained through analogy, constructionism calls for thinking about computers more like finger paint and less like a television. Computers, in other words, should be used for designing and creating, not for receiving:

> Teachers cannot simply pour information into the heads of learners; rather, learning is an active process in which people construct new understandings of the world around them through active exploration... people don't "get" ideas, they "make" them. (Italics in original) (Mitchel Resnick, 2002, p. 33)

To summarize, the foundation of a microworlds approach to learning is a constructionist, and an entailed constructivist, theory of knowledge acquisition. Learners construct their own understanding of the world by exploring it, and knowledge acquisition is particularly effective when they are making something meaningful within in it. The focus of education technologists is no longer on improving "teaching" to maximize the receipt of knowledge (as instructionism would have it). Instead, it is about setting up the right conditions for "learning": creating an environment for exploring, creating, and doing. In Papert's (1980b) own words, constructionism is about "giving children good things to 'do' (Italics in original), so that they can learn by doing much better than they could before". This raises the question of what exactly is a good thing to give children to do. For Papert, the answer to this question resides in a well-designed microworld.

What is a Microworld?

So what exactly is a microworld? In his key text *Mindstorms*, Papert (1980) provided a useful starting point when discussing his LOGO Turtle Graphics microworld:

> The Turtle World was a microworld, a "place", a "province of Mathland", where certain kinds of mathematical thinking could hatch and grow with particular ease. The microworld was an incubator... The design of the microworld makes it a "growing place" for a specific species of powerful ideas or intellectual structures. (Papert, 1980, p. 125)

Papert's choice of words here are very important. He used words like "hatch", "grow", and "species". This is because that for Papert, the microworld provided a means for learners to acquire knowledge in a "natural" way. Early language development in infants does not require didactic instruction: Language develops naturally through immersion within a linguistic environment. A microworld, for Papert, is an environment within which ideas in any domain can develop in a similar way. Turtle Graphics, for example, is a microworld where mathematical ideas can develop naturally. He referred to this natural process of learning, without the need for didactic instruction, as "Piagetian learning".

Papert's choice of words in the previous quotation is important for another reason. Words like "grow" and "hatch" are mixed in with words like "place" and "province of Mathland". This lexicon of place implies a need for borders or boundaries, which is a key part of a microworlds approach to learning. Put simply, microworlds should not be reduced to virtual sandpits that practice an "anything-goes" approach to learning. Microworlds are designed with boundaries to enable the discovery of key ideas within a particular domain of knowledge. As Mitchel Resnick puts it: "Microworlds are simplified worlds, specially designed to highlight (and make accessible) particular concepts and particular ways of thinking" (1997, p. 50).

More recently, Rieber (2004) has established a conceptual framework for understanding microworlds by framing them in terms of five functional attributes (p. 588). By discussing each of these attributes in turn, it is possible to gain a clearer picture of what a contemporary microworld may look like in practice.

The first attribute identified by Rieber is that microworlds are domain-specific. The seminal microworld of Turtle Graphics was designed so that learners discover powerful ideas in the domain of mathematics. In particular, Turtle Graphics was designed to cultivate an understanding in the area of geometry (Papert, 1972). By entering the directional command RIGHT 90, the learner will find that an on-screen turtle rotates around its centre through an angle of 90 degrees. One of the very most basic directional commands in Turtle Graphics, then, enables learners to explore the impact of increasing and decreasing the angle of turn.

Rieber's second attribute of a microworld is that it must be comprehensible to the learner. The microworld must have a low floor to facilitate access for learners at various stages of development. One of the key problems with the Turtle Graphics microworld is that learners are required to remember syntax. As Resnick et al. (2009) explained: "Early programming languages were too difficult to use, and many children simply couldn't master the syntax of programming" (p. 63). The Lifelong Kindergarten Groups at MIT have tried to lower the floor further by developing a building-block programming language called "scratch". Put simply, scratch allows the learner to piece together chunks of a program in an experience just like slotting together Lego bricks. Learners can build their own programs without having to remember the correct sequence of commands.

Third, the microworld must be intrinsically motivating for the learner. Learners must remain engaged with the microworld for long enough so that they could develop through stages of increasing complexity. If Rieber's second requirement was for a low floor, then it must also motivate to reach a high ceiling. This idea is similar to Rieber's fourth characteristic of a microworld as an agent for immersive and playful activity. One of the key design criteria for the Lifelong Kindergarten Group when creating Scratch was that it supported "many different types of projects (stories, games, animations, and simulations), so people with widely varying interests are all able to work on projects they care about" (Rieber, 2004, p. 64). If projects have a personal appeal, learners are more likely to immerse themselves within it. They will view the project positively as playing rather than negatively as work. This immersion will enable learners to gradually increase the complexity in their creations and facilitate learning within a particular domain of knowledge.

Rieber's final attribute of a microworld is that it should be designed and implemented with a constructivist paradigm of learning in mind. As Rieber reminded us that, this final attribute also carries a set of pedagogical assumptions external to the microworld itself (2004, p. 588). So when will a microworld no longer become a microworld? The answer is when students experience it through a series of didactic teacher-led tasks instead of learner-focused exploration. In *Mindstorms*, Papert (1980) set out his vision for a constructionist approach to learning with microworlds:

> The ... teacher will answer questions, provide help if asked, sometimes sit down next to a student and say “Let me show you something”. What is shown is not dictated by a set syllabus. (p. 179)

In summary, a microworld provides an environment for exploratory, Piagetian learning without didactic instruction. A microworld does, however, have some constraints: It should be specific to a particular domain of knowledge, and it needs a low floor (to enable easy access at first) and a high ceiling (to enable progression to increasing complexity later). Above all, the microworld should enable the learner to make something that is personally meaningful to them. With this checklist in mind, it is now possible to see the theory of microworlds put into practice.

Microworlds in Practice

As an example of a microworld in practice, the author would like to tell readers about a microworld project that he is going to be piloting in the secondary school where he teaches. It will be piloted with a group of 11-year-old children in their first year at secondary school. For their discrete ICT (information and communication technology) lessons, these children are been placed in a smaller support group, so that the pace of work can be suitably adjusted. All of the learners in the group have ALN (additional learning needs) and several have been issued statements of SEN (special educational need) by the local education authority. The children in the group benefit from a much higher LSA (learning support assistant)-to-student ratio than mainstream classes.

The project grew out the author’s interest in three educational tools: Scratch (the building-block programming language), LEGO Education WeDo (a real-world construction kit with sensors and motors), and PicoBoards (electronic sensor boards that provide inputs for the computer). By connecting the WeDo LEGO bricks and the Picoboard to the computer, Scratch is able to interact with the real-world through motors and sensors to provide a rich, exploratory experience for students.

Scratch has dramatically lowered the skill level required for beginners by removing the problem of syntax. The author’s microworld project, however, grew out of his desire to lower this floor even further. He wanted to increase access to learners with ALN, at least initially, so that they could get started on their programming creations more quickly and easily. In looking for ways to do this, the author became aware of the BYOB (build your own blocks) project.

As the name suggests, BYOB is an offshoot of Scratch that enables users to build their own blocks to extend the standard Scratch offering. The BYOB Website provides a rationale for the project “to extend the brilliant accessibility of Scratch to somewhat older users—in particular, non-CS-major computer science students—without becoming inaccessible to its original audience” (Monig & Harvey, 2011). The rationale of BYOB fitted perfectly with what the author was aiming to achieve, but in reverse, BYOB wanted to “extend” Scratch whereas the author wanted to “simplify” it. BYOB wanted to retain the standard elements of Scratch for its younger audience whilst the author wanted to retain them to extend the more-able learners within the support group. With this in mind, the author set out to use BYOB to design a new, simplified set of blocks. The author’s aim was simple to increase accessibility for the support group, so that they could experiment more easily with the motors and sensors that were communicating with Scratch. He started designing a microworld, or more accurately a microworld within a microworld (the author’s blocks are created in BYOB which is in turn based on Scratch), for the ALN students to use.

To illustrate by means of an example, the author will examine one block that he has created a little more

closely. Imagining a cartoon animal on the screen that is always in one of the two states, feeling dizzy or feeling normal, and depending on whether the input from the computer indicates an upright position. When the sprite is in a dizzy state, the sprite's costume needs to change to look dizzy. But if the sprite is not in a dizzy state, then it must be in a normal state, and the sprite's costume must change to reflect this. When using the standard set of Scratch programming blocks, the coding for this is not straightforward for beginners (see Figure 1). First, a sensing block is required to return the input from the tilt sensor. This must be placed within an operator block where the required amount of tilt is specified. This in turn forms the argument for the conditional block which encloses it. Finally, a loop surrounds the conditional, and the true or false values to return need to be placed in the named middle gaps.

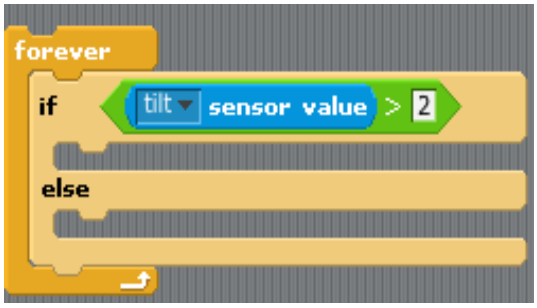

Figure 1. Standard Scratch tilt sensor blocks.

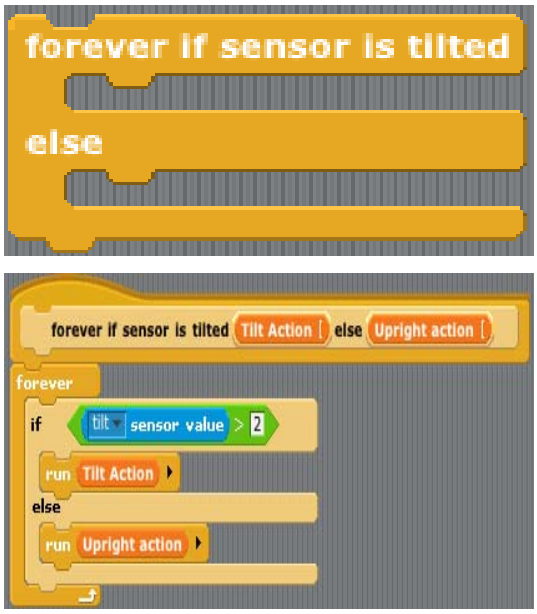

Figure 2. Student and teacher view of BYOB tilt sensor blocks.

To make working with sensors easier, these five blocks can be amalgamated into just one (see Figure 2) using the BYOB framework. The student needs only to add in the true or false values to be returned by the conditional. The numeric data type of the tilt sensor value have in effect been converted to Boolean, because the student now only has to deal with two potential inputs: tilted and not tilted. When more sophisticated control of the tilt input is required, students may progress to a model similar to that of the standard Scratch blocks (see Figure 1).

Many other simplified blocks can be created using BYOB in order to lower the floor of Scratch and to

make it more accessible to ALN learners. Now that the on-screen animal looks dizzy if you move him around (via tilt sensor inputs), the student may want him to make a scared noise if he hears a bang (via sound sensor inputs). As well as being in one of two states of dizzy or normal, the on-screen animal can now additionally be in one of the two states of startled and not-startled (see Figures 3 and 4). The animal could even be dizzy and startled at the same time, leading to a very perplexed animal indeed.

The microworld that is being created here through BYOB is still very much a project in development. The author is currently in the early stages of the project, and he is completing his microworld designs. A detailed study will be produced as part of his Ph.D. thesis to report on the findings of its implementation in the classroom.

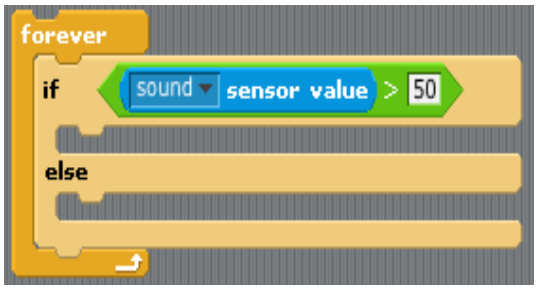

Figure 3. Standard Scratch sound sensor blocks.

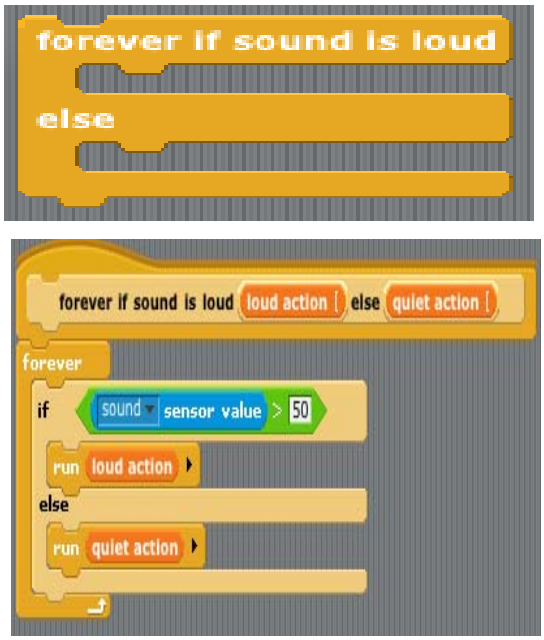

Figure 4. Student and teacher view of BYOB sound sensor blocks.

By exploring and experimenting within this microworld, it is hoped that learners are able to learn about some key concepts in computer science and control without becoming overwhelmed. The two simple blocks that the author has created here, for example, enables learners to find out about input variables, feedback loops and conditionals. It is hoped that design within this microworld enables learners to construct mental models of key computing concepts that will become vital later on in the education cycle. By experimenting with microworlds, learners are able to build powerful ideas in STEM domains.

References

Boden, M. (1994). *Piaget* (2nd ed.). London: Fontana.

Monig, J., & Harvey, B. (2011). *Build your own blocks*. Retrieved December 27, 2011, from http://byob.berkeley.edu/

Papert, S. (1972). A computer laboratory for elementary schools. *Computers and Automation, 21*(6), 19-23.

Papert, S. (1980a). *Mindstorms: Children, computers, and powerful ideas*. New York: BasicBooks.

Papert, S. (1980b). *Constructionism vs. instructionism*. Retrieved December 25, 2011, from http://www.papert.org/articles/ const_ inst/const_inst1.html

Papert, S. (1991). Situating constructionism. In I. Harel, & S. Papert (Eds.), *Constructionism* (pp. 1-11). Norwood, N. J.: Ablex.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York: BasicBooks.

Resnick, M. (1997). *Turtles, termites and traffic jams: Explorations in massively parallel microworlds*. Massachusetts: MIT.

Resnick, M. (2002). Rethinking learning in the digital age. In G. Kirkman (Ed.), *The global information technology report: Readiness for the Networked world* (pp. 32-37). Oxford: Oxford University Press.

Resnick, M., Maloney, J., Monrey-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60-67.

Rieber, L. (2004). Microworlds. In D. Donassen (Ed.), *Handbook of research for educational communications and technology* (2nd ed., pp. 583-603). Mahwah, N. J.: Lawrence Erlbaum Associates.

Taylor, R. P. (1980). Introduction. In R. P. Taylor (Ed.), *The computer in school, tutor tool, tutee* (pp. 1-10). New York: Teachers College Press.